TP4 – Estimation par l'algorithme EM

Exercice 1 : estimation des paramètres d'une ellipse

Dans ce TP, on cherche à estimer les paramètres d'une ellipse passant « au plus près » d'un ensemble de points, ces points formant un nuage dont la forme rappelle effectivement celle d'une ellipse. Le script `donnees_1` affiche une ellipse caractérisée par cinq paramètres réels tirés aléatoirement : la demi-longueur a du grand axe ; l'excentricité e ; les coordonnées (x_C, y_C) du centre ; enfin, l'angle polaire θ du grand axe. Ce script affiche également n_{app} points P_i situés au voisinage de l'ellipse, qui forment un ensemble d'apprentissage $\mathcal{D}_{\text{app}}$.

Pour estimer les paramètres recherchés, il est opportun de manipuler l'équation cartésienne d'une ellipse :

$$\alpha\, x^2 + \beta\, x\, y + \gamma\, y^2 + \delta\, x + \epsilon\, y + \phi = 0 \tag{1}$$

car cette équation est linéaire vis-à-vis du sextuplet de coefficients $(\alpha, \beta, \gamma, \delta, \epsilon, \phi)$. Ces six paramètres ne sont pas indépendants, puisque l'ellipse ne change pas s'ils sont tous multipliés par un même coefficient. En effet, comme nous l'avons déjà vu avec le vecteur de paramètres $\mathbf{p} = [a, e, x_C, y_C, \theta]$, une ellipse ne possède que cinq degrés de liberté. Les équations (1), écrites pour les n_{app} points de $\mathcal{D}_{\text{app}}$, forment un système linéaire *homogène* :

$$\mathbf{A}\, \mathbf{X}^\top = \mathbf{O} \tag{2}$$

où $\mathbf{X} = [\alpha, \beta, \gamma, \delta, \epsilon, \phi]$, $\mathbf{A}$ est une matrice de taille $n_{\text{app}} \times 6$, et $\mathbf{O}$ désigne le vecteur nul de $\mathbb{R}^6$. Afin d'éviter la solution triviale $\widehat{\mathbf{X}} = \mathbf{O}^\top$, on doit ajouter une contrainte. On peut par exemple imposer $\alpha + \gamma = 1$, car une ellipse vérifie forcément $\beta^2 - 4\,\alpha\,\gamma < 0$, donc il est impossible que $\alpha + \gamma = 0$. Cette contrainte ajoute une équation linéaire *non homogène* au système (2), qui n'admet donc plus la solution triviale $\widehat{\mathbf{X}} = \mathbf{O}^\top$.

Écrivez la fonction `MC`, appelée par le script `exercice_1`, visant à résoudre le système (2) complété par la contrainte $\alpha + \gamma = 1$, au sens des moindres carrés ordinaires. Le rôle de la fonction `calcul_score` est de mesurer la précision de l'estimation. Pour une paire d'ellipses passées en entrée, cette fonction calcule le rapport entre l'aire de l'intersection et l'aire de l'union. Le score calculé est compris entre 0 (lorsque les deux ellipses sont disjointes) et 1 (lorsque les deux ellipses sont identiques). Vérifiez que le score obtenu par `exercice_1` est maximal (égal à 1) lorsque l'écart-type σ du bruit sur les données d'apprentissage est égal à 0.

Une autre méthode d'estimation classique est celle du *maximum de vraisemblance*. Cette méthode très intuitive revient à répéter, pour un certain nombre d'ellipses tirées aléatoirement, le calcul d'une valeur, appelée *vraisemblance*, qui mesure l'adéquation des données d'apprentissage à cette ellipse. Une fois les vraisemblances calculées pour toutes les ellipses tirées aléatoirement, l'ellipse qui maximise la vraisemblance est retenue. La vraisemblance $L_{\mathbf{p}}(\mathcal{D}_{\text{app}})$ d'une ellipse de paramètres $\mathbf{p} = [a, e, x_C, y_C, \theta]$, relativement aux données $\mathcal{D}_{\text{app}}$, s'écrit :

$$L_{\mathbf{p}}(\mathcal{D}_{\text{app}}) = \prod_{i=1}^{n_{\text{app}}} f_{\mathbf{p}}(P_i) \tag{3}$$

où la densité de probabilité $f_{\mathbf{p}}(P_i)$ des points $P_i \in \mathcal{D}_{\text{app}}$ peut être modélisée par une loi normale :

$$f_{\mathbf{p}}(P_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{r_{\mathbf{p}}(P_i)^2}{2\,\sigma^2} \right\} \tag{4}$$

Dans cette expression, $r_{\mathbf{p}}(P_i)$ mesure l'adéquation du point P_i à l'ellipse de paramètres $\mathbf{p}$. Si C désigne le centre de l'ellipse, nous définissons $r_{\mathbf{p}}(P_i)$ comme la distance de P_i au point d'intersection entre l'ellipse et la droite (CP_i). Quant à σ, bien que ce paramètre soit également inconnu, nous supposons pour simplifier qu'il est connu et égal à l'écart-type du bruit sur les données d'apprentissage.

Comme un produit est plus difficile à maximiser qu'une somme, et que la fonction logarithme est strictement croissante, il est préférable de maximiser la *log-vraisemblance* $\ln L_{\mathbf{p}}(\mathcal{D}_{\text{app}})$. On doit donc résoudre le problème :

$$\max_{\mathbf{p}} \left\{ \ln \prod_{i=1}^{n_{\text{app}}} f_{\mathbf{p}}(P_i) \right\} = \min_{\mathbf{p}} \left\{ \sum_{i=1}^{n_{\text{app}}} r_{\mathbf{p}}(P_i)^2 \right\} \tag{5}$$

Dans (5), l'exponentielle de (4) a disparu grâce au logarithme, la maximisation est devenue une minimisation à cause signe $-$ dans l'argument de l'exponentielle, et les coefficients de normalisation ont été retirés, dans la mesure où ils ne changent rien à ce problème d'optimisation.

Écrivez la fonction `MV`, appelée par le script `exerice_1`, qui résout le problème (5) en calculant la somme $\sum_{i=1}^{n_{\text{app}}} r_{\mathbf{p}}(P_i)^2$ pour `nb_tirages` valeurs du vecteur de paramètres $\mathbf{p}$ tirées aléatoirement. Attention : inutile de coder le calcul de $r_{\mathbf{p}}(P_i)$, car ce calcul est fait par la fonction `calcul_r`, qui vous est fournie.

La précision de l'estimation par maximum de vraisemblance dépend de la valeur de `nb_tirages`. Néanmoins, pour une valeur de `nb_tirages` induisant un temps de calcul « raisonnable », cette deuxième estimation est généralement moins précise que celle aux moindres carrés. Elle permettra quand même de contribuer à la résolution d'un problème nettement plus compliqué, à savoir l'estimation des paramètres d'une paire d'ellipses.

Exercice 2 : estimation des paramètres d'une paire d'ellipses

Le script `donnees_2` affiche deux ellipses dont les paramètres sont tirés aléatoirement, ainsi que n_{app} points tirés aléatoirement au voisinage de chacune de ces ellipses, qui forment donc un ensemble d'apprentissage $\mathcal{D}_{\text{app}}$ comportant $2\,n_{\text{app}}$ points. Si les données d'apprentissage étaient déjà « partitionnées », c'est-à-dire si les croix étaient affichées en deux couleurs différentes, ce nouveau problème d'estimation ne serait pas plus compliqué que celui de l'exercice 1. Cela n'étant pas le cas, le problème devient nettement plus compliqué, car on ne sait quels points P_i choisir pour constituer un système linéaire tel que (2).

En revanche, il est facile de généraliser l'estimation par le maximum de vraisemblance des paramètres d'une paire d'ellipses. En effet, nous pouvons modéliser la nouvelle densité de probabilité $f_{\mathbf{p}_1,\mathbf{p}_2}(P_i)$ par un mélange de deux lois normales, à hauteur de proportions π_1 et $\pi_2 = 1 - \pi_1$:

$$f_{\mathbf{p}_1,\mathbf{p}_2}(P_i) = \frac{\pi_1}{\sigma_1\sqrt{2\pi}} \exp\left\{ -\frac{r_{\mathbf{p}_1}(P_i)^2}{2\,{\sigma_1}^2} \right\} + \frac{\pi_2}{\sigma_2\sqrt{2\pi}} \exp\left\{ -\frac{r_{\mathbf{p}_2}(P_i)^2}{2\,{\sigma_2}^2} \right\} \tag{6}$$

La maximisation de la log-vraisemblance $\ln L_{\mathbf{p}_1,\mathbf{p}_2}(\mathcal{D}_{\text{app}})$ s'écrit, d'après (3) et (6) :

$$\max_{\mathbf{p}_1,\mathbf{p}_2} \left\{ \ln \prod_{i=1}^{n_{\text{app}}} f_{\mathbf{p}_1,\mathbf{p}_2}(P_i) \right\} = \max_{\mathbf{p}_1,\mathbf{p}_2} \left\{ \sum_{i=1}^{n_{\text{app}}} \ln \left[\frac{\pi_1}{\sigma_1} \exp\left\{ -\frac{r_{\mathbf{p}_1}(P_i)^2}{2\,{\sigma_1}^2} \right\} + \frac{\pi_2}{\sigma_2} \exp\left\{ -\frac{r_{\mathbf{p}_2}(P_i)^2}{2\,{\sigma_2}^2} \right\} \right] \right\} \tag{7}$$

Nous continuons de supposer, pour simplifier, que les écarts-types σ_1 et σ_2 sont connus et égaux à l'écart-type du bruit sur les données d'apprentissage. D'autre part, nous choisissons $\pi_1 = \pi_2 = 0,5$ car les données d'apprentissage ont été produites, à partir des deux ellipses, dans ces mêmes proportions.

Faites une copie de la fonction `MV`, de nom `MV_2`, que vous modifierez de manière à permettre au script `exercice_2` de résoudre le problème (7). À moins de fixer `nb_tirages` à une valeur tellement élevée que cela rendrait le temps de calcul prohibitif, vous constatez que les résultats sont décevants. Néanmoins, une fois le problème (7) résolu, on peut partitionner les données en deux classes $k = 1$ et $k = 2$, en associant chaque donnée d'apprentissage $P_i \in \mathcal{D}_{\text{app}}$ à l'ellipse la plus proche, ce qui revient à calculer :

$$\widehat{k}(P_i) = \underset{k=1,2}{\arg\max} \left\{ \frac{\pi_k}{\sigma_k} \exp\left\{ -\frac{r_{\mathbf{p}_k}(P_i)^2}{2\,{\sigma_k}^2} \right\} \right\} \tag{8}$$

Écrivez la fonction `calcul_probas`, appelée par `exercice_2`, qui calcule les deux valeurs de l'objectif de (8), pour chaque point $P_i \in \mathcal{D}_{\text{app}}$, et retourne ces valeurs dans une matrice de taille $2 \times n_{\text{app}}$. Une fois la partition effectuée, le script `exercice_2` estime aux moindres carrés les paramètres de l'ellipse associée à chaque classe, à l'aide la fonction `MC` de l'exercice 1. Vous constatez que, dans presque tous les cas, la nouvelle paire d'ellipses obtenue « colle » mieux aux données que la paire d'ellipses estimées par le maximum de vraisemblance, ce que confirment les scores. Or, cette estimation peut être encore nettement améliorée en utilisant l'algorithme EM.

Exercice 3 : algorithme EM (Espérance-Maximisation)

La figure du milieu affichée par le script `exercice_2` indique quel mélange de lois maximise la vraisemblance, parmi un ensemble fini de lois de mélange tirées aléatoirement. Or, la probabilité de « tomber pile » sur les paramètres optimaux est quasiment nulle. En revanche, les ellipses de la figure de droite sont généralement plus proches des données d'apprentissage que celles de la figure du milieu. Une idée consiste donc à « boucler », c'est-à-dire à utiliser les ellipses de la figure de droite pour définir un nouveau mélange de lois et mettre à jour la partition.

L'algorithme EM, qui est un algorithme très général, s'inspire de cette idée, à ceci près qu'il n'effectue pas une partition stricte des données. Les paramètres à estimer sont initialisés par le script `exercice_2`, puis l'algorithme EM répète en boucle les deux étapes suivantes :

- **Étape E** – Calcul des probabilités d'appartenance aux deux classes des points d'apprentissage $P_i \in \mathcal{D}_{\text{app}}$:

$$\mathcal{P}_k(P_i) = \frac{\frac{\pi_k}{\sigma_k}\,\exp\left\{-\frac{r_{\mathbf{p}_k}(P_i)^2}{2\,{\sigma_k}^2}\right\}}{\frac{\pi_1}{\sigma_1}\,\exp\left\{-\frac{r_{\mathbf{p}_1}(P_i)^2}{2\,{\sigma_1}^2}\right\} + \frac{\pi_2}{\sigma_2}\,\exp\left\{-\frac{r_{\mathbf{p}_2}(P_i)^2}{2\,{\sigma_2}^2}\right\}}, \qquad k = 1, 2 \tag{9}$$

- **Étape M** – Mise à jour des proportions du mélange :

$$\pi_k = \frac{1}{n_{\text{app}}} \sum_{i=1}^{n_{\text{app}}} \mathcal{P}_k(P_i), \qquad k = 1, 2 \tag{10}$$

  puis estimation des paramètres des deux ellipses par résolution de deux systèmes de *moindres carrés pondérés*, qui comportent chacun n_{app} équations du type :

$$\mathcal{P}_k(P_i) \times \left(\alpha\, x_i^2 + \beta\, x_i\, y_i + \gamma\, y_i^2 + \delta\, x_i + \epsilon\, y_i + \phi\right) = 0$$

Faites une copie de la fonction `calcul_probas`, de nom `calcul_probas_EM`, et une copie de la fonction `MC`, de nom `MC_ponderes`, que vous modifierez de manière à réaliser, respectivement, les étapes E et M de cet algorithme. La plupart du temps, le script `exercice_3` parvient à trouver la « bonne » paire d'ellipses. Selon vous, pourquoi n'y parvient-il pas toujours ?